Constructors in Derived Classes:

1. Simple Inheritance:

class A {

public :

A ( ) { }

};

class B : public A {

public :

B ( ) { }

};

…

B objB;

When both the derived and base classes contain constructors, the base constructor is executed first and then the constructor of derived class is executed.

1. Multi-level Inheritance:

class A { A ( ) { } };

class B : public A { B ( ) { } };

class C : public B { C ( ) { } };

class D : public C { D ( ) { } };

…..

B objB;

C objC;

D objD; // First A ( ), B ( ), C ( ) and D ( )

1. Multiple Inheritance

class A { public : A ( ) { } };

class B { public : B ( ) { } };

class C : public B, public A { public : C ( ) { } }; // First B’s constructor, then A’s and then C’s

In case of multiple inheritance, the base classes are constructed in the order in which they appear in the declaration of the derived class.

Destructor

class A { A( ) { } ~A( ) { } };

class B : public A { B ( ) { } ~B( ) { } };

…

{

B objB;//First B’s destructor, then A’s

}

Nesting of Classes

class A { members…

class B { members … };

};

Dynamic Memory:

new: Allocates memory at run-time within the heap for the variable

Syntax : new data-type;

delete:

new data-type [ size ];

delete [ ] address;

2 x 5 = 10bytes

\_ \_ \_ \_ \_

5000 5002 5004 5006 5008

Arr[0]

5000 + (0x2)

Pointers to objects:

Of type class

this pointer

Pointers to Derived Class

Pointers to objects of a base class are type-compatible with pointers to objects of a derived class.

class Base{

…

};

class Derived : public Base{

…

};

Base objB;

Derived objD;

Base \*ptrB;

ptrB = &objB;

ptrB = &objD;

virtual functions: // Late-Binding

virtual returnType funName( ){

…

}

…

….

returnType funName ( ){

…

}

Pure Virtual Functions:

Abstract Base Class: